REMARKS

Claims 1-26 and 29-38, all the claims pending in the application, stand rejected on prior art grounds. Claims 1-7 stand rejected upon informalities. Moreover, claim 6 is objected to because of informalities, and the specification is objected to. Claims 1, 6-8, 13-20, 25, 29, and 34 are amended herein. Moreover, the specification is amended herein. Applicants respectfully traverse these rejections based on the following discussion.

I. The Objection to the Specification

The specification is objected to because it contains an embedded hyperlink on page 2, line 9. Accordingly, the Applicants have amended the specification to remove the hyperlink from the text. Additionally, the Applicants have amended the specification to cure spelling/grammatical errors. In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw this objection.

II. The Objection to the Claims

Claim 6 is objected to because of a misspelling. Accordingly, the Applicants have amended claim 6 to change "affininty" to "affinity". In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw this objection.

III. The 35 U.S.C. §112, Second Paragraph, Rejection

Claims 1-7 stand rejected under 35 U.S.C. §112, second paragraph. These rejections are traversed as explained below. Claim 1 (and claims 2-7 by dependency) are rejected because

there is insufficient antecedent basis for the claimed language. Accordingly, the Applicants have amended claim 1 to change "said intermediate form UPC-unique constructs" to "first intermediate form". Claim 5 (and claims 6-7 by dependency) are rejected because the Office Action indicates there is insufficient antecedent basis for "said UPC-unique constructs" in line 1 of claim 5. However, this language is provided several times in independent claim 1 (from which claims 5-7 depend). Accordingly, the Applicants respectfully contend that there is sufficient antecedent basis for "said UPC-unique constructs" in claim 5. In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw these rejections.

IV. The Prior Art Rejections

Claims 1-26 and 29-38 stand rejected under 35 U.S.C. §102(e) as being anticipated by Odani, et al. (U.S. Patent No. 6,760,906 B1), hereinafter referred to as Odani. Applicants respectfully traverse these rejections based on the following discussion.

The Applicants strongly but respectfully disagree that the Applicants' claimed invention is anticipated by Odani. The two works are independent and orthogonal and teach different improvements in the art of translating/compiling source-level programs. As prior art, the Applicants' specification states in the first paragraph of its background section that "A compiler generally includes a frontend that translates source code into an intermediate form. An intermediate form processor optimizes the intermediate form. A backend then converts the optimized intermediate form code into, typically, machine language code". The Applicants' specification cites A. V. Aho, R. Sethi, and J. D. Ullman, "Compilers Principles, Techniques, and Tools", Addison-Wesley Publishing Company, Reading, Massachusetts, 1988, as a textbook

that teaches such prior art. Odani falls into this arrangement of prior art as shown its figures 2 and 10 wherein a compiler front end is invoked followed by a parallelizer module followed by an object code generator. Odani teaches improvements in the art of translating/compiling source-level programs via its parallelizer module wherein it processes and optimizes the intermediate form generated by the front-end. This is stated by Odani in column 1, lines 13-18, "and more particularly relates to code optimization technology specially designed for a parallel processor."

However, it is noteworthy and patentably distinct that Odani does not refer to Unified Parallel C, UPC, or UPC constructs and does not teach any method of translating UPC-specific constructs to object code. Furthermore, the model of assembler-instruction-level parallelism taught by Odani and cited by Odani as evident in prior art VLIW processor architectures is very different from the language-level parallelism expressed in a language such as UPC, which is comprised of independent threads on multiple machines/processors cooperating and computing with each other via a non-uniform shared memory space. A parallel computation in a UPC program is user specified, with constructs such as UPC memory references translated into communication library calls. This is very different from parallelization analyses in VLIW architectures and Odani which seek to re-organize instructions at the assembly code level into words or execution units whose component instructions execute in parallel within a machine or processor. Furthermore, the parallelization in Odani is automatic - it is not user-specifiable at the source program level.

Additionally, in contrast to Odani, the Applicants' invention is UPC specific and teaches simplified methods of code generation for UPC constructs along with an efficient method of translating the UPC forall construct. The Applicants' invention concludes its background section

with a paragraph pointing out the need for a compiler to further ease the code generation aspects of translating UPC programs and in particular handle the UPC forall statements: "The need exists, however, for a compiler to further ease the code generation aspects of translating UPC programs and to correctly handle UPC program constructs. This is particularly so for nested forall statements". The first paragraph of the summary section then points out that code generation takes place in the form of C-level code strings for UPC constructs to be spliced into the source UPC program: "The gist of the invention is to generate C-level code strings for UPC constructs to be spliced into the source UPC program". As the Applicants' conclusion section argues in its second paragraph, this inventive step as opposed to code generation directly in an intermediate form achieves the benefits that: "The solution is able to reuse the compiler vendor's frontend code extensively, for translating the (generated) C strings to intermediate form and for taking care of optimization information. The automation in intermediate code generation that is able to be obtained thus reduces the complexity faced in digesting UPC annotations substantially. For example, all of the (extensive) initialization code for UPC data types is expressed completely within C strings in the disclosed method and no intermediate code needs to be generated/manipulated directly for this purpose. The overall result is a minimally modified standard C compiler that is also capable of translating UPC programs to executable object codes". These features are further described in the Applicants' claims.

The standard, prior art C compiler technology that is reused comprises a frontend module that is invoked in a two-iteration loop as shown in Figures 1 and 2. In the first iteration, Figure 2 steps 50, 52 and 54, the C-level code strings for UPC data types are generated. In the second iteration, the generated C code strings are fed back to the frontend module to be spliced into the

original UPC program followed by a front-end invocation that translates the C-strings automatically into intermediate code (Figure 2, step 62). Thus, frontend functionality of translating source code into intermediate code is used twice (Figure 2, steps 52 and 62). This two-iteration loop of Figures 1 and 2 is distinct from the non-looping and standard arrangement of compiler components in Figures 2 and 10 of Odani.

The Office Action states, "as per claim 1, Odani discloses a method for compiling unified parallel c-language (UPC) source code containing UPC-unique constructs and c-language constructs (e.g. Fig. 2, element 10 and related text)." Figure 2, element 10 is described on column 8, line 60 – column 9, line 2 of Odani. It is also described in conjunction with element 10 of figure 10 on column 12, lines 18-21. The element is a compiler front-end that takes source code written in a high-level language like C and generates intermediate code for it divided into basic blocks. Since Odani does not refer to unified parallel C or UPC directly, the application of element 10, Figure 2 to UPC syntax implies the application of prior art UPC technology in element 10, such as the Berkeley UPC Compiler technology cited in the Applicants' background section. Compared to such UPC prior art, the Applicants' invention offers a beneficial inventive step of simplified code generation for UPC constructs and a better treatment of UPC forall loops.

Next, the Office Action indicates that Odani teaches, "translating said UPC source code into a first intermediate form (in column 5, lines 25-30 "... translating source code into intermediate code ...")." As discussed above, this translation into a first intermediate form using prior art UPC compiler technology loses the code generation benefit of the two-step loop of the Applicants' claimed invention.

Next, the Office Action indicates that Odani teaches, "generating proxy form C-language

code strings of data components within said intermediate form UPC-unique constructs (e.g. Fig. 4 and related text)." Fig. 4 illustrates exemplary internal form code received by the basic block boundary parallelizer included in the program processor shown in Fig. 2 (Odani's column 5, lines 63-65). This is intermediate or assembler-level code (column 8, lines 63-64) and is not source-level code written in the C language. It is also not proxy form code pertaining to UPC-unique constructs in intermediate form.

Next, the Office Action provides that Odani teaches, "inserting said generated code strings into said UPC source code to form a combined code (in column 1, lines 25-31 "... source program and then combines them into a single long ...")." The instructions extracted by the compiler are assembler code or intermediate-form level instructions representing the source program and these instructions are packed into parallel executable words. One of ordinary skill in the art understands that a "word" is not a syntactic or semantic construct in higher-level languages like C catered to by this VLIW technology, so representation of source-level code in parallel executable words is not possible within the source language. Also, this model of intra-machine or intra-processor parallelism at the instruction level is very different from the multiple software processes/threads and communication via shared memory model of UPC that operates on multiple machines/processors. The combined code constructed by the claimed sub-step of the Applicants' invention above represents a valid UPC source code since UPC is a parallel dialect of C and the additional code added to a UPC program is pure C code, which continues to maintain the original UPC source code as a valid UPC code after transformation.

Next, the Office Action states that Odani teaches, "translating said combined code into a second intermediate form, wherein any statements within said UPC-unique constructs are placed

in a C-form with associated program text *(in column 11, lines 20-25 "... translates the internal form code ...")*, and surviving UPC-unique constructs are discarded (in column 9, lines 17-23 "...and removes the execution boundary ..." and Fig. 6 and related text)." However, the text in *italics* above is not self-consistent – the cited column and lines do not contain what the Office Action suggests it teaches. The notion of execution boundaries of parallel-executable sets of instructions on a processor is orthogonal/independent/irrelevant to the notion of thread-level parallelism enshrined in UPC. For instance, the discarding of surviving UPC-unique constructs corresponds to dead-code elimination which does not affect the parallel execution model of the UPC program, while removing the execution boundary as discussed in the context of Figure 6, Odani, actually changes the set of instructions that are executed in parallel on a processor.

Next, the Office Action indicates that Odani teaches, "converting said second intermediate form to compiled machine code (in column 19, lines 50-55 "... converting the intermediate ..." and e.g. Fig. 2, element 20 and related text)." This conversion of intermediate code referred to in Odani is to parallel execution units. This is very different from the generation of object code from intermediate code.

Next, the Office Action states that Odani teaches, "as per claim 2, Odani discloses the method of claim 1, wherein code strings are proxy declarations (in column 22, lines 25-30 "... instructions are ... variable ...")." However, there is no notion of proxy data structures or declarations in Odani. In fact, the notion of variable units containing instructions to be executed in parallel is independent to and immaterial to the Applicants' invention.

Next, the Office Action indicates that Odani teaches, "as per claim 3, Odani discloses the method of claim 2, wherein a said proxy declaration includes a name that is a mangled version of

a name of a respective UPC-unique data component having a one-to-one mapping (e.g. Fig. 17 and related text)." Again, there is no notion of proxy data structures or declarations, or mangled versions of names in Odani. In fact, Fig. 17 is a dependence graph (see Odani, column 6, lines 30-32), which has nothing to do with the Applicants' invention.

Next, the Office Action indicates that Odani teaches, "as per claim 4, Odani discloses the method of claim 1, wherein said associated program text includes a conditional statement (e.g. Fig. 13, element S2 and related text and in column 13, lines 60-65 "... if there are several instructions ..."). " However, claim 4 refers to the presence of a conditional statement in translated intermediate form code as opposed to a conditional in the decision-making process (e.g. Odani's Fig. 13, element S2 and other cited material).

Next, the Office Action states, "as per claim 5, Odani discloses the method of claim 4, wherein said UPC-unique constructs are forall statements, and said associated program text includes a conditional statement whose predicates leads to evaluation based upon an affinity test (e.g. Fig. 13, element S2 and related text and in column 13, lines 60-65 "... if there are several instructions ..."). " Applicants' claim 5 refers to the presence of forall statements in UPC source constructs and that the translated intermediate form code contains a conditional whose predicate evaluation carries out an affinity test. This is quite different from the presence of a conditional in the decision-making process (e.g. Odani's Fig. 13, element S2 and other cited material).

Next, the Office Action indicates that Odani teaches, "as per claim 6, Odani discloses the method of claim 5, wherein, for forall statements having an affinity other than "continue" (e.g. Fig. 13, element S2 and related text), the translation step includes sub-traversal of a forall body and determining the context of each static level of nesting (e.g. Fig. 12 and related text)."

However, there is nothing in Odani that teaches forall processing. Figure 13, element S2 is a decision-making node in a distinct flowgraph and Figure 12 is a dependence graph (Odani's column 6, lines 17-18).

Next, the Office Action indicates that Odani teaches, "as per claim 7, Odani discloses the method of claim 6, further comprising the step of incrementing a depth variable in accordance with each said sub-traversal (e.g. Fig. 12 and related text)." As mentioned above, there is nothing in Odani that teaches forall processing or its details.

In view of the foregoing, the Applicants respectfully submit that the collective cited prior art do not teach or suggest the features defined by amended independent claims 1, 8, 13, 20, 29, and 34 and as such, claims 1, 8, 13, 20, 29, and 34 are patentable over Odani. Further, dependent claims 2-7, 9-12, 14-19, 21-26, 30-33, and 35-38 are similarly patentable over Odani, not only by virtue of their dependency from patentable independent claims, respectively, but also by virtue of the additional features of the invention they define. Thus, the Applicants respectfully request that these rejections be reconsidered and withdrawn. Moreover, the Applicants note that all claims are properly supported in the specification and accompanying drawings. In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw the rejections.

V. Formal Matters and Conclusion

With respect to the objections/rejections to the specification and claims, the specification and claims have been amended, above, to overcome these rejections. In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw the objections/rejections to the specification and claims.

In view of the foregoing, Applicants submit that claims 1-26 and 29-38, all the claims presently pending in the application, are patentably distinct from the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary. Please charge any deficiencies and credit any overpayments to Attorney's Deposit Account Number 09-0441.

Respectfully submitted,

Dated: February 16, 2007

Mohammad S. Rahman
Registration No. 43,029

Gibb I.P. Law Firm, LLC
2568-A Riva Road, Suite 304
Annapolis, MD 21401
Voice: (301) 261-8625
Fax: (301) 261-8825
Customer Number: 29154